УДК 519.688

Об одном методе сокрытия информации в GIF файлах

Строкин Д.И., Пономарев И.В.
Алтайский государственный университет, г. Барнаул
mega.strokin@mail.ru, igorpon@mail.ru

Аннотация

За последние несколько десятилетий, в связи с повсеместным развитием информационных технологий и средств мультимедиа, значительную актуальность приобретает разработка новых методов хранения, передачи, анализа и воспроизведения данных. К числу таких методов также относятся средства обеспечения надёжности, защищённости, безопасности и конфиденциальности информации при её передаче по различным каналам связи.

В данной работе рассматриваются методы шифрования информации с помощью методов компьютерной стеганографии. Изучаются способы встраивания защищаемой информации в GIF файлы. Приводятся конкретные алгоритмы реализации полученных процедур.

Ключевые слова: стеганографическая система, формат GIF, контейнер, палитра, индекс цвета, наименее значащий бит.

1. Введение

В настоящий момент построение высокозащищенных систем требует применения технологий, базирующихся на использовании методов компьютерной стеганографии. Задачи скрытной передачи информации, встраивания лицензионных данных в объекты цифрового контента, аутентификации пользовательских данных и камуфлирования программного обеспечения – всё это области применения стеганографии. Например, задачи защиты авторского права по отношению к цифровому контенту уже достаточно давно решаются с помощью технологии цифровых водяных знаков (ЦВЗ).

Стеганосистема – это совокупность средств и методов, которые используются для формирования скрытого канала передачи информации.

Основные положения, которые необходимо учитывать при построении стеганосистем [1]:

- стеганосистема должна иметь приемлемую вычислительную сложность реализации;
- противник имеет полное представление о стеганографической системе и деталях ее реализации. Единственной информацией, которая остается неизвестной потенциальному противнику, является ключ, с помощью которого только его держатель может установить факт присутствия и содержание скрытого сообщения;
- если противник каким-то образом узнает о факте существования скрытого сообщения, это не должно позволить ему извлечь подобные сообщения в других данных до тех пор, пока ключ хранится в тайне;
- потенциальный противник должен быть лишен каких-либо технических и иных преимуществ в распознавании или раскрытии содержания тайных сообщений.

2. Основные принципы компьютерной реализации

На сегодняшний день существует достаточное количество программных решений, использующих медиафайлы, привычные обычному пользователю, для передачи зашифрованного сообщения: видеофайлы (форматов MPEG и AVI), аудиофайлы (форматов MP3 и WAV), изображения (форматов BMP, JPEG, JPEG2000, PNG, GIF). Многие из них также позволяют передавать достаточно большой объем скрываемой информации, сохраняя при этом высокий уровень качество используемого контейнера и возможность их использования по основному назначению.

Все такие программные решения базируются на двух принципах:

1. файлы, содержащие оцифрованное изображение или звук, могут быть до некоторой степени видоизменены без потери функциональности, в отличие от других типов данных, требующих абсолютной точности;

2. неспособность органов чувств человека различить незначительные изменения в цвете изображения или качестве звука, что особенно легко использовать применительно к объекту, несущему избыточную информацию, будь то 16-битный звук, 8-битное или, еще лучше, 24-битное изображение.

Базовым подходом к реализации методов компьютерной системы в рамках той или иной информационной среды лежит выделение малозначительных фрагментов этой среды и замена существующей в них информации другой информацией, которую необходимо скрыть. Поскольку в компьютерной стеганографии рассматриваются среды, поддерживаемые средствами вычислительной техники и компьютерными сетями, то вся информационная среда в результате может быть представлена в цифровом виде.

Формат .GIF, имеющий блочную структуру, поддерживает блоки с комментариями, которые никак не влияют на конечное изображение, но содержат дополнительную информацию. Сокрытие сообщений в данных блоках нельзя считать полноценным стеганографическим методом, так как конфиденциальная информация может быть обнаружена любым текстовым или шестнадцатеричным редакторами.

Далее будем рассматривать алгоритмы, использующие в качестве стегоконтейнеров изображения – статичные и не статичные. Данный выбор обусловлен следующими причинами [1]:

- большим объёмом пространства сокрытия, то есть участков, то есть участков, в которых система может скрыть информацию;

- заранее известным размером контейнера;

- наличием в большинстве изображений текстурных областей, имеющих шумовую структуру и хорошо подходящих для встраивания информации;

- слабой чувствительностью человеческого глаза к незначительным изменениям цветов изображения, его яркости, контрастности, содержанию в нем шума, искажения вблизи контуров;

- задачей защиты фотографий, картин, видео от незаконного тиражирования и распространения;

- широкой распространенностью файлов-изображений в сети интернет.

3. Стеганографические методы, использующие наименее значащие биты изображения

Один из наиболее распространенных методов сокрытия информации в пространственной области изображений – метод замены Наименее Значащего Бита (Least Significant Bit), основной принцип которого заключается в том, что передаваемая информация встраивается в значения младших битов изображения. Модификация именно таких битов не способна восприниматься человеческим зрением, так как они несут в себе меньше всего информации [2].

Суть метода заключается в следующем: Допустим, имеется 8-битное изображение в градациях серого. 00 (00000000) обозначает чёрный цвет, FF (11111111) – белый. Всего имеется 256 градаций. Также предположим, что сообщение состоит из 1 байта – например, 01101011. При использовании 2 младших бит в описаниях пикселей, нам потребуется 4 пикселя. Допустим, они чёрного цвета. Тогда пиксели, содержащие скрытое сообщение, будут выглядеть следующим образом: 00000001 00000010 00000010 00000011.

Рассмотрим две вариации данного метода, так или иначе основывающиеся на особенности выбранного формата – использовании палитры. Нам известно, что нельзя напрямую менять пиксели изображения, так как каждый пиксель – это индекс в таблице цветов. Элементы, близкие по индексу, могут иметь совершенно разные представления в цветовом пространстве, и поэтому, изменения младшего бита могут привести к заметным изменениям самого изображения.

I) Наиболее простым решением будет менять значения самих цветов палитры.

Каждый элемент палитры кодируется 24 битами по 8 бит на каждую компоненту RGB, таким образом, в один элемент мы способны внедрить до 3-4 бит сообщения, при этом не вызывая заметных для человеческого зрения графических аномалий.

Однако, очевидно, что длина сообщения будет ограничена размером палитры, например, для палитры из 256 различных цветов максимальная длина сообщения – 1024 бита (или 128 байт) [2].

Данный недостаток можно нивелировать использованием локальных палитр, но в таком случае упадёт стеганографическая стойкость метода.

II) Другой метод, использующий значения младших бит изображения, основывается на малой разности цветовых интенсивностей некоторых элементов палитры [2]. Например, значения яркости для цветов (255, 255, 255) и (255, 254, 253) будут некритично отличаться, а значит, индекс одного элемента можно легко заменить индексом другого.

Для этого цвета в палитре сортируют по возрастанию веса W и в уже отсортированной палитре выбираются пары элементов, для которых разность весов W меньше заданной пороговой величины d. Обозначим одну такую пару за (j_i, j_k), где i и k – это индексы элементов в неотсортированной палитре, причем в отсортированной таблице j_i от j_k отличается на 1.

Сокрытие сообщения состоит в том, что последовательно просматриваются все точки изображения, по значению точки k определяется соответствующий номер j_k. Если элемент j_k пригоден для сокрытия, то его НЗБ заменяется на очередной бит сообщения. Затем по получившемуся номеру $j_{k'}$ определяется связанный с ним элемент исходной таблицы k', который и присваивается текущей точке.

Извлечение сообщения будет происходить аналогичным способом:
Последовательно просматриваются все точки изображения. Для текущей точки k ищется номер j_k в отсортированной по весу W палитре цветов и если:

- Младший бит индекса j_k равен нулю, смотрим, удовлетворяет ли пара $(j_k, j_k + 1)$ условию: $W_{j_k+1} - W_{j_k} < d$. Если удовлетворяет, значит, из индекса j_k извлекаем младший бит и записываем его в сообщение.

- Младший бит индекса j_k равен единице, смотрим, удовлетворяет ли пара $(j_k - 1, j_k)$ условию: $W_{j_k} - W_{j_k-1} < d$. Если удовлетворяет, значит, из индекса j_k извлекаем младший бит и записываем его в сообщение.

В качестве веса можно использовать следующие характеристику

$$W = R \cdot 65536 + G \cdot 256 + B.$$

4. Структура GIF. Построение декодера

Вследствие того, что в сети практически отсутствуют программы и библиотеки, позволяющие редактировать данные GIF-изображений, перед созданием рабочей стеганографической системы и реализацией исследуемых методов, необходимо построить полноценный декодер данного формата.

Структура изображений GIF, как правило, содержит следующие блоки [3, 4]:

1. Дескриптор логического экрана – содержит данные, необходимые для определения области отображения изображения на экране устройства. Обычно к данному дискриптору приписывают заголовок формата – «GIF89a» или «GIF87a».

   Занимает 13 байт и выглядит следующим образом:

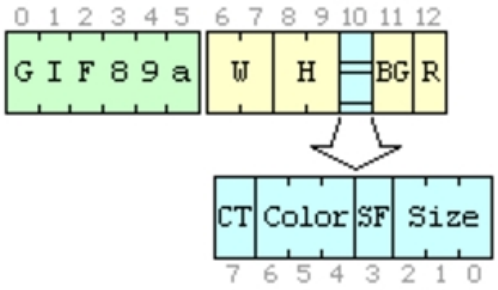

Рисунок 1. Дескриптор логического экрана

   На рисунке 1: W, H – ширина и высота логического экрана в пикселах; 11-ый байт отвечает за наличие и размер глобальной палитры, глубину цвета изображения; флаг CT – отвечает за наличие глобальной палитры цветов; Color – задает цветовое разрешение; SF – флаг, отвечающий за сортировку таблицы по значимости (частоте использования) цветов; Size – число цветов в палитре; BG – номер цвета фона; R – соотношение сторон изображения.

2. Глобальная палитра цветов – идёт сразу за дескриптором логического экрана (если используется). Имеет размер $3 \cdot 2^{Size+1}$.

Рисунок 2. Схема палитры

3. Блок расширения приложения (обычно Netscape) – специальный блок, с которым может работать только приложение, для которого оно предназначено. Имеет размер 19 байт.

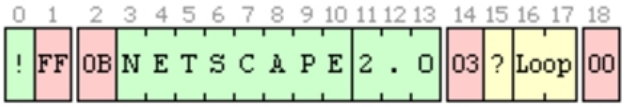

Рисунок 3. Блок расширения палитры

На рисунке 3: ! – специальный символ, означающий начало блока расширения; FF (255) – байт, означающий начало расширения приложения; 0B (11), 03 – размеры субблоков данных; Loop – количество циклов анимации; 00 – терминатор блока.

4. Блок расширения управления графикой – общее описание анимации, действующее на следующий после расширения графический блок. Размер – 8 байт.

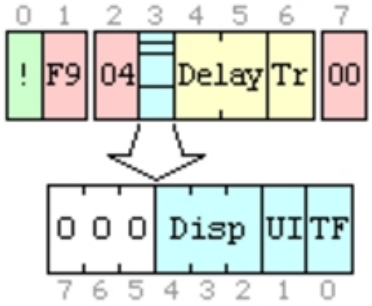

Рисунок 4. Блок расширения графикой

На рисунке 4: ! – специальный символ, означающий начало блока расширения; F9 (249) – байт, означающий начало расширения управления графикой; 04 – размер субблока данных; 4-ый байт практически не используется, кроме флага прозрачности TF; Delay – время задержки кадра на экране в 1/100 секунд; Tr – задает номер прозрачного цвета, если флаг TF равен единице; 00 – терминатор блока.

5. Дескриптор изображения – обязательный блок, задающий параметры для следующего после него изображения (кадра). Размер – 10 байт.

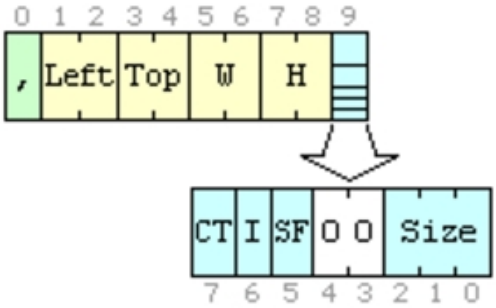

Рисунок 5. Дескриптор изображения

На рисунке 5: , (44) – специальный символ, означающий начало дескриптора нового изображения; Left, Top – положения изображения на логическом экране по вертикали и горизонтали, соответственно; W, H – ширина и высота изображения в пикселах; 9-ый байт отвечает за наличие и размер локальной палитры; флаг CT – отвечает за наличие локальной палитры цветов; I – использование чересстрочной развертки изображения; SF – флаг, отвечающий за сортировку таблицы по значимости (частоте использования) цветов; Size – число цветов в палитре.

6. Графические блоки – непосредственно, данные очередной картинки GIF-изображения, сжатые по алгоритму LZW. Состоит из отдельных субблоков по 255 байт в каждом (за исключением последнего).

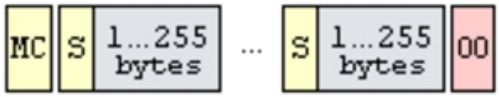

Рисунок 6. Схема графического блока

На рисунке 6: MC – начальный размер LZW-кода, обычно равен глубине цвета картинки; S – размер субблока данных. Размер субблоков должен быть равен 255 байтам. У последнего субблока (или если он вообще один) размер может быть любым – от 1 до 255 байт, после S идут байты от 1 до 255 – данные изображения.

Таблица 1

Определение начального размера LZW-кода

MC	Размера LZW-кода
8	9
7	8
6	7
5	6
4	5
3	4
2	3

5. Алгоритмы сжатия и распаковки LZW

Кодирование данных происходит следующим образом [4]:

1. Инициализируем словарь по количеству цветов палитры + 2 специальных кода: clear и end. Кода clear очищает словарь индексов; код end сигнализирует о конце блока данных.

2. Из потока индексированных цветов берется символ и добавляется к текущей строке символов. Полученная строка сравнивается со всеми строками в словаре, и, если такая строка уже присутствует, повторяем шаг 2. Иначе переходим к шагу 3.

3. Если мы обнаружили, что текущая строка символов отсутствует в словаре, то добавляем ее на первое свободное место, а в поток данных записываем индекс последней строки, совпавшей с текущей. Текущая строка обнуляется, снова переходим к шагу 2.

Ниже приведен псевдокод сжатия методом LZW:

```
ИНИЦИАЛИЗАЦИЯ СЛОВАРЯ
СТРОКА = очередной символ из входного потока
WHILE входной поток не пуст DO
СИМВОЛ = очередной символ входного потока
IF СТРОКА + СИМВОЛ IN СЛОВАРЬ THEN
СТРОКА = СТРОКА + СИМВОЛ
```

```
ELSE
вывести в выходной поток номер для СТРОКА
IF размер словаря > 4064
заново инициализировать СЛОВАРЬ
вывести в выходной поток код для CLEAR
END of IF
вывести в выходной поток код для СТРОКА
добавить в таблицу строк СТРОКА+СИМВОЛ
СТРОКА = СИМВОЛ
END of IF
END of WHILE
вывести в выходной поток код для СТРОКА
вывести в выходной поток код для END
```

Декодирование данных [4]:

1. Алгоритм, как и в случае сжатия, начинается с инициализации словаря по количеству цветов палитры + 2 специальных кода: clear и end.

2. Далее считывается очередной код (обозначим за C) из массива данных, сжатых методом LZW. Ищем его среди элементов словаря: если нашли, переходим к шагу 3. Иначе переходим к шагу 4.

3. Если найденный код соответствует: коду Clear – заново инициализируем словарь; если коду End – декомпрессия данных завершена. Если код C не соответствует ни одному из этих символов, то переходим к шагу 4.

4. Если переменная «предыдущий код» (обозначим за P) задана, сравниваем значение считанного кода C с размером словаря. Если очередной код C равен размеру словаря, считываем строку индексов (обозначим за S) по «предыдущему коду» P, иначе – по считанному коду C. В словарь добавляем новый элемент, строка символов которого равна считаной строке S + первый элемент из S. В выходной массив индексированных цветов заносим строку S. Приравниваем P = C и переходим к шагу 2.

Псевдокод распаковки:

```
ИНИЦИАЛИЗАЦИЯ СЛОВАРЯ
WHILE входной поток не пуст DO
НОВЫЙ КОД = очередной код из входного потока
IF НОВЫЙ КОД = CLEAR THEN
заново инициализировать СЛОВАРЬ
END of IF
IF НОВЫЙ КОД = END THEN
EXIT
END of IF

IF НОВЫЙ КОД = размер СЛОВАРЯ THEN
СТРОКА = считать строку из словаря по номеру СТАРЫЙ КОД
ELSE
СТРОКА = считать строку из словаря по номеру НОВЫЙ КОД
END of IF
вывести в выходной поток СТРОКА
СИМВОЛ = первый символ СТРОКИ
```

```
добавить в СЛОВАРЬ СТРОКА+СИМВОЛ
СТАРЫЙ КОД = НОВЫЙ КОД
END of WHILE
```

6. Выводы

Разработанная программная реализация имеет приемлемую ресурсоемкость и сложность вычислений – общее время сокрытия сообщений зависит лишь от количества отдельных кадров GIF-изображения. В худшем случае, на кодирования одного кадра изображения уходит, примерно, 3 секунды.

Наибольшую пропускную способность показывают алгоритм, основанный на методе замены наименее значащих бит данных изображения. Достаточно высокую пропускную способность имеет алгоритм, использующий одинаковые элементы палитры. Наименьшую пропускную способность имеет метод замены младших битов элементов палитры.

Наименьшие визуальные воздействия на файл контейнер осуществляет метод замены наименее значащих бит данных изображения; а метод, основанный на наличии одинаковых элементов в палитре, напротив, иногда способен приводить к графическим аномалиям.

Список литературы

1. Грибунин В.Г., Оков И.Н., Туринцев И.В. Цифровая стеганография. — М. : СОЛОН-ПРЕСС, 2009.

2. Аргановский А.В., Балакин А.В., Грибунин В.Г., Сапожников С.А. Стеганография, цифровые водяные знаки и стеганоанализ. — М. : Вузовская книга, 2009.

3. пецификация формата GIF. GRAPHICS INTERCHANGE FORMAT(sm) Version 89a (c)1987,1988,1989,1990 Copyright CompuServe Incorporated Columbus, Ohio. 1990. — URL: https://www.w3.org/Graphics/GIF/spec-gif89a.txt.

4. Сэломон Д. Сжатие данных, изображений и звука. — М. : Техносфера, 2004.

5. Барсуков В.С., Романцов А.П. Компьютерная стеганография вчера, сегодня, завтра // Специальная техника. — 1998. — № 4,5. — С. 19–26.

6. Изычева А.В., Сидоренко В.Г. Стеганографические методы защиты информации. Учебное пособие. — М., 2009.

7. Конахович Г.Ф., Пузыренко А.Ю. Компьютерная стеганография. Теория и практика. — К. : «МК-Пресс», 2006.

8. Урбанович П.П. Защита информации методами криптографии, стеганографии и обфускации. — Минск : БГТУ, 2016.

9. Хорошко В.А., Чекатков А.А. Методы и средства защиты информации. — К. : Юниор, 2003.

10. Shannon C.E. The communication theoryof secrecy systems // Bell Sys. Tech. J. — 1949. — Т. 28, № 4.

11. Westfeld A., Pfitzmann A. Attacks on Steganographic Systems. Breaking the Steganographic Utilities EzStego, Jsteg, Steganos and S-Tools and Some Lessons Learned // Lecture Notes in Computer Science. — 2020. — 1768:61–75.